US 6519268 corresponds

PD 030009

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Übersetzung der europäischen Patentschrift**

(97) **EP 0 885 418 B1**

(10) **DE 697 03 732 T 2**

(51) Int. Cl.[7]: **G 06 F 13/12**

(21) Deutsches Aktenzeichen: 697 03 732.0
(86) PCT-Aktenzeichen: PCT/US97/02546
(96) Europäisches Aktenzeichen: 97 906 666.9
(87) PCT-Veröffentlichungs-Nr.: WO 97/33230
(86) PCT-Anmeldetag: 19. 2. 1997
(87) Veröffentlichungstag der PCT-Anmeldung: 12. 9. 1997
(97) Erstveröffentlichung durch das EPA: 23. 12. 1998
(97) Veröffentlichungstag der Patenterteilung beim EPA: 20. 12. 2000
(47) Veröffentlichungstag im Patentblatt: 13. 6. 2001

22264 U.S.PTO 10/761529 012004

DE 697 03 732 T 2

(30) Unionspriorität:
612321 07. 03. 1996 US

(73) Patentinhaber:
Sony Electronics Inc., Park Ridge, N.J., US

(74) Vertreter:
derzeit kein Vertreter bestellt

(84) Benannte Vertragstaaten:
AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(72) Erfinder:
SMYERS, D., Scott, San Jose, US

(54) ASYNCHRONES DATENÜBERTRAGUNGSGERÄT ZUR VERWALTUNG ASYNCHRONER DATENÜBERTRAGUNGEN ZWISCHEN EINEM ANWENDUNGSPROGRAMM UND EINER BUSSTRUKTUR

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Di Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Mark namt inhaltlich nicht geprüft.

DE 697 03 732 T 2

GEBIET DER ERFINDUNG

Die vorliegende Erfindung betrifft das Gebiet der automatischen Verwaltung von Datenübertragungsoperationen zwischen einer Anwendung und einer Busstruktur. Im Besonderen betrifft die vorliegende Erfindung das Gebiet des automatischen Erzeugens von Transaktionen, die erforderlich sind, um eine asynchrone Datenübertragungsoperation zwischen einer Anwendung und einer Busstruktur abzuschließen.

STAND DER TECHNIK

Der Standard IEEE 1394 "P1394 Standard For A High Performance Serial Bus", Fassung 8.01v1, vom 16. Juni 1995, ist ein internationaler Standard für die Implementierung einer kostengünstigen Hochgeschwindigkeits-Architektur für einen seriellen Bus, der sowohl asynchrone als auch isochrone Datenübertragungsformate unterstützt. Bei isochronen Datenübertragungen handelt es sich um Echtzeitübertragungen, die so erfolgen, dass die Zeitintervalle zwischen Kennzeitpunkten bei sendenden und empfangenden Anwendungen die gleiche Dauer aufweisen. Jedes isochron übertragene Datenpaket wird in einem eigenen Zeitraum übertragen. Ein Beispiel für eine ideale Anwendung für die isochrone Datenübertragung ist die Übertragung von einem Videorekorder zu einem Fernsehgerät. Der Videorekorder zeichnet Bilder und Ton auf und speichert die Daten in diskreten Blöcken oder Paketen. Der Videorekorder überträgt daraufhin jedes Paket, das Aufzeichnungen von Bild und Ton über einen begrenzten Zeitraum darstellt, zur Anzeige durch das Fernsehgerät. Die Busarchitektur gemäß IEEE 1394 Standard sieht mehrere Kanäle für die isochrone Datenübertragung zwischen Anwendungen vor. Ein sechs Bit Kanalnummer wird mit den Daten übermittelt, um den Empfang

BEST AVAILABLE COPY

durch die entsprechende Anwendung zu gewährleisten. Dies ermöglicht es, dass mehrere Anwendungen gleichzeitig isochrone Daten über die Busstruktur übertragen. Asynchrone Übertragungen sind herkömmliche Datenübertragungsoperationen, die so schnell wie möglich erfolgen, und bei denen eine Datenmenge von einer Quelle an einen Empfänger bzw. an ein Ziel übertragen wird.

Der IEEE 1394 Standard sieht einen seriellen Hochgeschwindigkeits-Bus zur Verbindung digitaler Vorrichtungen miteinander vor, wodurch eine universale E/A-Verbindung vorgesehen wird. Der IEEE 1394 Standard definiert eine digitale Schnittstelle für die Anwendungen, wodurch es nicht erforderlich ist, dass eine Anwendung digitale Daten in analoge Daten umsetzt, bevor diese über den Bus übermittelt werden. Somit empfängt eine empfangende Anwendung auch digitale Daten von dem Bus und keine analogen Daten, und somit muss diese Anwendung auch keine analogen Daten in digitale Daten umsetzen. Das für den IEEE 1394 Standard erforderliche Kabel weist im Vergleich für größere Kabel für derartige Vorrichtungen einen sehr dünnen Durchmesser auf. Einem IEEE-1394-Bus können während der Bus aktiv ist, Vorrichtungen hinzugefügt bzw. von diesem entfernt werden. Wenn eine Vorrichtung hinzugefügt oder entfernt wird, konfiguriert sich der Bus automatisch neu, um Daten zwischen den zu diesem Zeitpunkt vorhandenen Knoten zu übermitteln. Ein Knoten wird als ein logisches Objekt mit eindeutiger Adresse an der Busstruktur angesehen. Jeder Knoten sieht einen Kennungs-ROM, eine standardisierte Anordnung von Steuerregistern und einen eigenen Adressraum vor.

Der IEEE-1394-Standard definiert ein Protokoll gemäß der Abbildung aus Figur 1. Das Protokoll weist einen seriellen

BEST AVAILABLE COPY

Bus-Verwaltungsblock 10 auf, der mit einer Transaktionsschicht 12 gekoppelt ist, eine Verknüpfungsschicht 14 und eine physikalische Schicht 16. Die physikalische Schicht 16 sieht die elektrische und mechanische Verbindung zwischen einer Vorrichtung oder einer Anwendung und dem IEEE 1394-Kabel vor. Die physikalische Schicht 16 sieht ferner die Arbitrierung vor, die gewährleistet, dass alle mit dem IEEE 1394-Bus verbundenen Vorrichtungen Zugriff auf den Bus haben, wobei ferner die tatsächliche Übermittlung und der Empfang von Daten vorgesehen werden. Die Verknüpfungsschicht 14 sieht den Datenpaket-Übermittlungsdienst für die Übertragung asynchroner und isochroner Datenpakete vor. Dies unterstützt sowohl die asynchrone Datenübertragung, unter Verwendung eines Quittungsprotokolls, als auch die isochrone Datenübertragung, wobei ein garantiertes Bandbreiten-Protokoll in Echtzeit für eine Just-in-Time-Datenübermittlung vorgesehen wird. Die Transaktionsschicht 12 unterstützt die erforderlichen Befehle für den Abschluss asynchroner Datenübertragungen, darunter Lesen, Schreiben und Sperren. Der serielle Bus-Verwaltungsblock 10 weist einen isochronen Betriebsmittel-Manager zur Verwaltung isochroner Datenübertragungen auf. Der serielle Bus-Verwaltungsblock 10 sieht ferner eine Steuerung der Gesamtkonfiguration des seriellen Busses in Form einer Optimierung des Arbitrierungs-Timings vor, der Gewährleistung ausreichender elektrischer Leistung für alle Vorrichtungen an dem Bus, die Zuweisung des Zyklus-Masters, die Zuweisung isochroner Kanal- und Bandbreiten-Ressourcen sowie grundlegende Fehlermitteilungen.

Zur Initialisierung einer isochronen Übertragung kann es sein, dass mehrere asynchrone Datenübertragungen erforderlich sind, um die Anwendungen zu konfigurieren und um den bestimmten Kanal zu bestimmen, der für die Datenübermittlung verwendet

BEST AVAILABLE COPY

wird. Sobald der Kanal bestimmt worden ist, werden Puffer an der übermittelnden Applikation bzw. Anwendung verwendet, um die Daten vor dem Senden zu speichern, und wobei Puffer an der empfangenden Applikation bzw. Anwendung dazu verwendet werden, die Daten vor deren Verarbeitung zu speichern. Bei einigen peripheren Implementierungen ist es wünschenswert, dass das Peripheriegerät große Datenmengen unter Verwendung einer großen Anzahl asynchroner Transaktionen überträgt. Um diese Transaktionen schnell und effizient zu erzeugen, ist es ungünstig, wenn eine Allzweck-CPU oder ein Mikrocontroller für die Gestaltung jedes angeforderten Pakets erforderlich ist.

In EP-A-042811 wird ein asynchroner Betriebsmodus unter Verwendung eines Controllers mit direktem Speicherzugriff (DMA-Controller) und eines Ein-Chip-Mikrocomputers mit einer Mehrzahl von Adressregistern offenbart, die sich jeweils aus einem Quelladressenregister zum Speichern einer Quelladresse und einem Zieladressenregister zum Speichern einer Zieladresse zusammensetzen. Die beschriebene Anordnung setzt den Eingriff einer CPU voraus. In US-A-4.493.021 wird ein lokales Netzwerk (LAN) für eine Mehrzahl autonomer Computer beschreiben, die mit verschiedenen Geschwindigkeiten bzw. Frequenzen und unter verschiedenen Protokollen arbeiten. Ein Hostrechner unterteilt eine zu übermittelnde Nachrichtendatei in Blöcke, die jeweils einen Header mit einem Datentyp-Bezeichner und einen Trailer aufweisen. Ein zugeordneter Netzwerkbus, der die Computer mit einem globalen Bus verbindet, weist Netzwerkadapter auf, welche die Daten in Pakete unterteilen, die jeweils einen Header aufweisen, dem ein Transport-Header und ein Trailer hinzugefügt werden, wobei ein Rahmentyp-Code einen von drei Modi der Adressierung bei der Datenübermittlung spezifiziert, d.h. einen physikalischen Adressmodus Übermittlungen von Computer zu Computer unter Verwendung von zwei Bytes für

BEST AVAILABLE COPY

Quell- und Zieladressen, einen logischen Adressmodus und einen Datentyp-Modus.

Benötigt wird ein asynchroner Datenübertragungskanal, der eine automatische Erzeugung von Transaktionen vorsieht, die für den Abschluss einer asynchronen Datenübertragungsoperation erforderlich ist, ohne dass eine Überwachung durch eine Programmierschnittstelle (API) und den Prozessor einer Applikation erforderlich ist.

ZUSAMMENFASSUNG DER ERFINDUNG

Vorgesehen ist gemäß einem ersten Aspekt der vorliegenden Erfindung ein asynchroner Datenübertragungskanal gemäß dem gegenständlichen Anspruch 1.

Vorgesehen ist gemäß einem zweiten Aspekt der vorliegenden Erfindung ein Verfahren zur Verwaltung einer Schreib-Datenübertragungsoperation gemäß dem gegenständlichen Anspruch 10.

Vorgesehen ist gemäß einem dritten Aspekt der vorliegenden Erfindung ein Verfahren zur Verwaltung einer Lese-Datenübertragungsoperation gemäß dem gegenständlichen Anspruch 15.

Ein asynchroner Datenübertragungskanal (ADP) erzeugt automatisch die erforderlichen Transaktionen zum Abschluss asynchroner Datenübertragungsoperationen für eine Anwendung über eine Busstruktur. Der asynchrone Datenübertragungskanal weist eine Registerdatei auf, die durch die Anwendung bzw. die Applikation programmiert wird. Die Registerdatei ermöglicht der Anwendung das Programmieren von Anforderungen und

BEST AVAILABLE COPY

Merkmalen für die Datenübertragungsoperation. Die Registerdatei umfasst die Busgeschwindigkeit, das Transaktionsetikett, den Transaktionscode, den Zielknotenbezeichner, einen Paketzähler, in Paketzähler-Erhöhungsfeld, ein Steuerfeld und ein Zustandsfeld. Nach dem die Registerdatei durch die Anwendung programmiert und eingeleitet worden ist, erzeugt der asynchrone Datenübertragungskanal automatisch die Lese- oder Schreib-Transaktionen, die erforderlich sind, um die Datenübertragungsoperation über den entsprechenden Adressbereich abzuschließen, und zwar unter Verwendung der Informationen in der Registerdatei als Mustervorlage für die Erzeugung der Transaktionen und Header. Der asynchrone Datenübertragungskanal erhöht automatisch den Wert in dem versetzten Zieladressfeld für jede Transaktion gemäß der Länge jedes Datenpakets, sofern das Erhöhungsmerkmal nicht deaktiviert worden ist, wodurch angezeigt wird, dass die Transaktionen an einer einzigen Adresse zu erfolgen haben. Der Paketzählerwert stellt die Anzahl der verbleibenden zu erzeugenden Transaktionen dar. Der Paketzählerwert wird nach der Übertragung jedes Pakets herabgesetzt bzw. erniedrigt. Das Paketzähler-Erhöhungsfeld ermöglicht der Anwendung die Erhöhung des Paketzählerwertes durch Schreiben in das Paketzähler-Erhöhungsfeld.

Mehrere asynchrone Datenübertragungskanäle können in einem System zur Verwaltung mehrerer asynchroner Datenübertragungsoperationen vorhanden sein. Bei einem derartigen System weist jeder asynchrone Datenübertragungskanal einen eindeutigen Transaktionsetikettwert oder -wertebereich auf. Ein Multiplexer ist mit jedem asynchronen Datenübertragungskanal gekoppelt, um die Transaktionen und Datenpakete von den

BEST AVAILABLE COPY

asynchronen Datenübertragungskanälen auf die Busstruktur zu multiplexen. Ein Demultiplexer ist ebenfalls mit jedem asynchronen Datenübertragungskanal gekoppelt, um Signale und Datenpakete von der Busstruktur zu empfangen und diese zu dem entsprechenden asynchronen Datenübertragungskanal zu leiten, und zwar unter Verwendung des Transaktionscodes und der Transaktionsetikettenwerte.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

Es zeigen:

Figur 1 ein durch den IEEE 1394 Standard definiertes Protokoll;

Figur 2 ein schematisches Blockdiagramm eines Link-Chips mit drei asynchronen Datenübertragungskanälen gemäß der vorliegenden Erfindung; und

Figur 3 eine Registerdatei in jedem asynchronen Datenübertragungskanal.

GENAUE BESCHREIBUNG DES BEVORZUGTEN AUSFÜHRUNGSBEISPIELS

Ein asynchroner Datenübertragungskanal gemäß der vorliegenden Erfindung erzeugt automatisch die erforderlichen asynchronen Transaktionen für die Implementierung asynchroner Datenübertragungen zu und von einer Anwendung bzw. einem Anwendungsprogramm über eine Busstruktur. Der Begriff Anwendung bzw. Applikation bezieht sich hierin entweder auf ein Anwendungsprogramm oder einen Gerätetreiber. Bei der Busstruktur, über die die Datenübertragungsoperationen ausgeführt werden, handelt es sich vorzugsweise um eine

BEST AVAILABLE COPY

Busstruktur gemäß dem IEEE 1394 Standard. Für den Fachmann ist diesbezüglich allerdings erkennbar, dass sich der erfindungsgemäße asynchrone Datenübertragungskanal auch zur Verwendung bei der Verwaltung von Datenübertragungen über andersartige Busstrukturen eignet. Der asynchrone Datenübertragungskanal ist in Richtung der Anwendung in der Lage, jede beliebige Datenmenge zwischen einem lokalen Datenpuffer oder einem durch die Anwendung vorgesehenen FIFO und einem Adressbereich unter Verwendung mehrerer asynchroner Transaktionen zu übertragen.

Der asynchrone Datenübertragungskanal weist eine Registerdatei auf, die durch die Anwendung programmiert wird, wenn eine Datenübertragungsoperation abgeschlossen werden soll. Die Registerdatei ermöglicht es der Anwendung, bestimmte Voraussetzungen für die Datenübertragungsoperation zu programmieren, darunter die Busgeschwindigkeit, mit der die Transaktionen erzeugt werden müssen, ein Transaktionsetikett und ein Transaktionscode, der die Art der Transaktion darstellt, einen Bezeichner für den Zielknoten, mit dem die Übertragung durchgeführt wird, eine versetzte Zieladresse, welche die Anfangsadresse darstellt, an der die Übertragung erfolgt, und eine Länge jedes Datenpakets. Die Registerdatei weist ferner einen Paketzähler auf, der dazu dient, die Anzahl der verbleibenden zu erzeugenden Pakete anzuzeigen, ein Paketzähler-Erhöhungsfeld, das der Anwendung die Erhöhung des Paketzählers ermöglicht, ein Steuerfeld und ein Zustandsfeld. Das Erhöhungsmerkmal des asynchronen Datenübertragungskanals kann durch die Anwendung ausgeschaltet werden, wenn die Transaktionen an der Busstruktur an einer einzigen Adresse erfolgen sollen.

BEST AVAILABLE COPY

Nach dem die Registerdatei durch die Anwendung programmiert und eingeleitet worden ist, erzeugt der asynchrone Datenübertragungskanal automatisch die erforderlichen Lese- oder Schreib-Transaktionen, um die Datenübertragungsoperation über den entsprechenden Adressbereich abzuschließen. Die Informationen in der Registerdatei werden von dem asynchronen Datenübertragungskanal als Mustervorlage für die Erzeugung der erforderlichen Transaktionen und der entsprechenden Header zur Beendigung der Datenübertragungsoperation verwendet. Der asynchrone Datenübertragungskanal erhöht automatisch den Wert in dem versetzten Zieladressfeld für jede Transaktion gemäß der Größe bzw. dem Umfang der übertragenen Pakete, sofern die Erhöhungsfunktion nicht deaktiviert worden ist. Da der asynchrone Datenübertragungskanal die erforderlichen Transaktionen automatisch erzeugt, ist keine unmittelbare Prozessorsteuerung oder Überwachung durch die einleitende Anwendung erforderlich. Dies ermöglicht der Anwendung die Ausführung anderer Funktionen sowie die Durchführung anderer Aufgaben, während der erfindungsgemäße asynchrone Datenübertragungskanal die Datenübertragungsoperation abschließt. Die Registerdatei weist jedoch das Paketzähler-Erhöhungsfeld auf, das es der Anwendung ermöglicht, die Anzahl der verbleibenden, durch den asynchronen Datenübertragungskanal durchzuführenden Transaktionen zu erhöhen. Auf diese Weise ist der asynchrone Datenübertragungskanal in der Lage, bei Bedarf die Erzeugung der erforderlichen Transaktionen für den Abschluss einer Datenübertragungsoperation zu regeln.

Ein System kann eine Mehrzahl asynchroner Datenübertragungskanäle zur Verwaltung einer Mehrzahl von asynchronen Datenübertragungsoperationen aufweisen. Bei einem derartigen System ist ein Multiplexer zwischen die Busstruktur

BEST AVAILABLE COPY

und jeden der asynchronen Datenübertragungskanäle geschaltet, um die Transaktionen und Datenpakete von dem asynchronen Datenübertragungskanal auf die Busstruktur zu multiplexen. Ferner ist ein Demultiplexer mit jedem asynchronen Datenübertragungskanal gekoppelt, um Signale und Datenpakete von der Busstruktur zu empfangen und diese zu dem entsprechenden asynchronen Datenübertragungskanal zu leiten. Der Demultiplexer verwendet den Transaktionscode und das Transaktionsetikett, um festzustellen, welcher asynchrone Datenübertragungskanal die Daten empfangen soll bzw. empfängt. Innerhalb des Systems weist jeder asynchrone Datenübertragungskanal einen eigenen eindeutigen Transaktionsetikettwert oder -wertebereich auf.

Eine Link-Schaltung bzw. Verknüpfungsschaltung mit drei asynchronen Datenübertragungskanälen (ADP) gemäß der vorliegenden Erfindung ist in der Abbildung aus Figur 2 dargestellt. In dem bevorzugten Ausführungsbeispiel ist die Link-Schaltung 10 auf einer einzigen integrierten Schaltung bzw. einem Chip ausgebildet. Die Link-Schaltung 10 sieht eine Verknüpfung zwischen den Anwendungen 12 und 14 und einer Busstruktur 58 vor. Die Anwendungen 12 und 14 sind beide mit einem Systembus 16 gekoppelt. Der Systembus 16 ist mit jedem der First-in-First-out-Datenpuffer (FIFOs) 32, 34 und 36 gekoppelt. Die Anwendungen 12 und 14 sind ferner beide mit einer Programmierschnittstellen-Schaltung 18 verbunden. Die Programmierschnittstellen-Schaltung 18 ist mit einer Reihe von Steuerregistern 38, mit jedem asynchronen Datenübertragungskanal 20, 22 und 24 und mit einem Link-Kern 44 verbunden. Jeder asynchrone Datenübertragungskanal 20, 22 und 24 weist eine entsprechende Registeranordnung 26, 28 und 30 auf. Jedes FIFO 32, 34 und 36 entspricht einem entsprechenden asynchronen Datenübertragungskanal 20, 22 und

BEST AVAILABLE COPY

24. Das FIFO 32 ist mit dem asynchronen Datenübertragungskanal 20 gekoppelt. Das FIFO 34 ist mit dem asynchronen Datenübertragungskanal 22 gekoppelt. Das FIFO 36 ist mit dem asynchronen Datenübertragungskanal 24 gekoppelt. Die Steuerregister 38 sind mit jedem asynchronen Datenübertragungskanal 20, 22 und 24 verbunden. Jeder asynchrone Datenübertragungskanal 20, 22 und 24 ist mit einem Multiplexer 40 für abgehende Datenübertragungsoperationen und mit einem Demultiplexer 42 für eingehende Datenübertragungsoperationen verbunden. Im Sinne dieser Offenbarung handelt es sich bei einer abgehenden Datenübertragung um eine Übertragung von einer Anwendung zu der Busstruktur, während es sich bei einer eingehenden Datenübertragung um eine Übertragung von der Busstruktur zu einer Anwendung handelt.

Der Link-Kern 44 weist einen Sender 46, einen Empfänger 48, einen Zyklus-Timer 50, einen Zyklus-Monitor 52, eine CRC-Fehlerprüfschaltung 54 (CRC = zyklische Redundanzprüfung) und eine physikalische Schnittstellenschaltung 56 als physikalische Schnittstelle zu der Busstruktur 58 auf. Der Sender 46 ist mit dem Multiplexer 40, dem Zyklus-Timer 50, der CRC-Fehlerprüfschaltung 54 und der physikalischen Schnittstellenschaltung 56 verbunden. Der Empfänger 48 ist mit dem Demultiplexer 42, dem Zyklus-Monitor 52, der CRC-Fehlerprüfschaltung 54 und der physikalischen Schnittstellenschaltung 56 verbunden. Der Zyklus-Timer 50 ist mit dem Zyklus-Monitor 52 gekoppelt. Die physikalische Schnittstellenschaltung 56 ist mit der Busstruktur 58 gekoppelt.

Das in der Abbildung aus Figur 2 dargestellte System weist drei asynchrone Datenübertragungskanäle 20, 22 und 24 auf. Für

BEST AVAILABLE COPY

den Fachmann ist ersichtlich, dass abhängig von den jeweiligen Anforderungen der Systeme ein System mit jeder beliebigen Anzahl von asynchronen Datenübertragungskanälen 20, 22 und 24 implementiert werden kann. Jeder asynchrone Datenübertragungskanal bietet die Möglichkeit zur automatischen Abwicklung einer Datenübertragungsoperation für eine Anwendung. Wie dies aus der folgenden Beschreibung deutlich wird, steigern zusätzliche asynchrone Datenübertragungskanäle in einem System die Fähigkeiten des Systems, indem gleichzeitig asynchrone Datenübertragungsoperationen ausgeführt werden können.

Jeder asynchrone Datenübertragungskanal stellt einen bidirektionalen Datenpfad für Daten zu und von einer Anwendung dar, die über asynchrone Transaktionen über die Busstruktur 58 übermittelt werden. Vor jede Operation eines Datenübertragungskanals muss ein externes Objekt eine Registerdatei in dem asynchronen Datenübertragungskanal programmieren. Bei dem externen Objekt kann es sich um die Anwendung selbst handeln oder um eine andere Intelligenz oder eine Zustandsvorrichtung innerhalb des Systems. In dem bevorzugten Ausführungsbeispiel der Erfindung wird die Registerdatei des asynchronen Datenübertragungskanals durch die Anwendung programmiert. Jeder asynchrone Datenübertragungskanal ist in der Lage, die erforderlichen Header für abgehende Daten zu erzeugen und die Header zu prüfen und daraus eingehende Daten zu entfernen, und zwar unter Verwendung der Registerdatei als Mustervorlage.

Die Registerdatei des asynchronen Datenübertragungskanals weist Daten auf, die sich auf die Anfangsadresse der Busstruktur, die Transaktionsart und die Transaktionsgröße beziehen, wie dies nachstehend im Text näher beschrieben wird.

BEST AVAILABLE COPY

In dem bevorzugten Ausführungsbeispiel kann es sich bei der Transaktionsart um eine der folgenden Optionen handeln: Quadlet-Lesen, Quadlet-Schreiben, Block-Lesen oder Block-Schreiben. Bei einer Quadlet-Transaktion beträgt die Transaktionsgröße vier Byte, und bei Blocktransaktionen entspricht die Größe der Blockanforderungsgröße.

Bei einer Freigabe überträgt der asynchrone Datenübertragungskanal Anwendungsdaten unter Verwendung asynchroner Transaktionen gemäß den in der Registerdatei programmierten Parameter. Im Falle von Schreib-Transaktionen von der Anwendung zu einem anderen mit der Busstruktur gekoppelten Knoten verwendet der asynchrone Datenübertragungskanal die an der FIFO-Schnittstelle verfügbaren Anwendungsdaten, überträgt die entsprechenden Header-Daten zu den Daten in dem von dem Link-Kern 44 vorausgesetzten Format und überträgt die Daten über den Multiplexer 40 zu dem Link-Kern 44. Bei Lese-Transaktionen von einem anderen mit der Busstruktur gekoppelten Knoten zu der Anwendung gibt der asynchrone Datenübertragungskanal die entsprechenden Leseanforderungspakete ab, und wenn die Daten empfangen werden, werden die Daten in den entsprechenden Leseantwortpaketen über die FIFO-Schnittstelle zu der Anwendung geleitet. Bei den Lese- und Schreib-Transaktionen organisiert der asynchrone Datenübertragungskanal die Daten in für die Busstruktur spezifische Paketformate gemäß den Anforderungen des Link-Kerns 44. Der asynchrone Datenübertragungskanal übernimmt ferner die Adressberechnung für Transaktionen für einen größeren Adressbereich, wie dies für die Anforderung der Anwendung erforderlich ist. Mit anderen Worten werden folgende Transaktionen bei einem größeren Adressbereich in dem Adressraum der Busstruktur adressiert.

BEST AVAILABLE COPY

Die FIFO-Schnittstelle für jeden asynchronen Datenübertragungskanal ist direkt mit einem FIFO 32, 34 oder 36 gekoppelt, das dem Datenpfad zugeordnet ist, den der asynchrone Datenübertragungskanal steuert. Jedes FIFO 32, 34 oder 36 ist einem einzelnen asynchronen Datenübertragungskanal zugeordnet. Die Link-Schnittstelle für jeden asynchronen Datenübertragungskanal ist über den Multiplexer 40 und den Demultiplexer 42 mit dem Link-Kern 44 gekoppelt. Die von jedem asynchronen Datenübertragungskanal dem Link-Kern 44 präsentierten Daten weisen ein Format auf, das von der Link-Kern-Funktion vorausgesetzt wird. Jeder asynchrone Datenübertragungskanal ist so gestaltet, dass er Daten von dem Link-Kern 44 in dem Format empfängt, das durch die Link-Kern-Spezifikation definiert ist. Wenn mehr als ein asynchroner Datenübertragungskanal in dem System vorgesehen ist, ist jeder asynchrone Datenübertragungskanal über den Multiplexer 40 und den Demultiplexer 42 mit dem Link-Kern 44 gekoppelt.

Die Daten von dem Link-Kern 44 zu den asynchronen Datenübertragungsleitungen 20, 22 und 24 werden durch den Demultiplexer 42 geleitet. Der Demultiplexer 42 verwendet den Transaktionscode und das Transaktionsetikett um die Daten zu dem entsprechenden asynchronen Datenübertragungskanal zu leiten. Der Demultiplexer 42 leitet Antwortpakete von der Busstruktur 58 zu dem entsprechenden asynchronen Datenübertragungskanal unter Verwendung des Transaktionscodefelds des Paket-Headers und des Werts in dem Transaktionsetikettfeld des Paket-Headers. Danach stimmt der asynchrone Datenübertragungskanal die Antwortpakete mit den entsprechenden Anforderungspaketen ab.

Der Demultiplexer 42 ändert keine Informationen, wenn er Pakete von dem Link-Kern 44 zu dem entsprechenden asynchronen

BEST AVAILABLE COPY

Datenübertragungskanal leitet. Alle durch den Link-Kern erzeugten Informationen werden zu dem asynchronen Ziel-Datenübertragungskanal geleitet. Der asynchrone Datenübertragungskanal führt alle erforderlichen Manipulationen der Daten von dem Link-Kern 44 durch bevor diese Daten zu der Anwendung übertragen werden, wobei dies ein etwaiges Entfernen von Header-Informationen gemäß den Erfordernissen des Protokolls für die Busstruktur beinhalten kann. Für abgehende Daten bereitet der asynchrone Datenübertragungskanal Daten von der Anwendung vor, so dass diese in dem entsprechenden, von dem Link-Kern 44 vorausgesetzten Format gegeben sind. Jeder asynchrone Datenübertragungskanal erzeugt die entsprechenden Header-Informationen und bettet diese in die Daten von der Anwendung ein, bevor die Daten über den Multiplexer 40 an den Link-Kern 44 gesendet werden.

Für alle asynchronen Datenübertragungskanäle 20, 22 und 24 erzeugt und verbraucht die Link-Schnittstelle Daten in einem Format, das mit den Anforderungen der Funktion des Link-Kerns 44 kompatibel ist. Während der Schreibeoperation erzeugen die asynchronen Datenübertragungskanäle 20, 22 und 24 die erforderlichen spezifischen Header-Informationen für die Busstruktur und betten diese in die Daten von der Anwendung gemäß den Erfordernissen des Link-Kerns 44 ein. Während einer Lese-Operation nimmt der asynchrone Datenübertragungskanal Daten in dem durch den Link-Kern 44 vorgesehenen Format für Daten an, die sich von dem Link-Kern 44 zu einem der asynchronen Datenübertragungskanäle 20, 22 und 24 bewegen. Mit anderen Worten ist keine Manipulation der Daten erforderlich, um Daten von dem Link-Kern 44 zu dem entsprechenden asynchronen Datenübertragungskanal 20, 22 oder 24 zu verschieben.

BEST AVAILABLE COPY

Wenn nur ein asynchroner Datenübertragungskanal in einem System vorhanden ist, kann der asynchrone Datenübertragungskanal direkt mit dem Link-Kern 44 verbunden werden. Wenn mehrere asynchrone Datenübertragungskanäle in einem System vorhanden sind, muss das System einen entsprechenden Multiplexer 40 und Demultiplexer 42 zwischen den asynchronen Datenübertragungskanälen und dem Link-Kern 44 aufweisen. Der Multiplexer 40 ist dafür verantwortlich, die Daten an den Link-Schnittstellen der Mehrzahl von asynchronen Datenübertragungskanälen 20, 22 und 24 zu erfassen und diese Daten in den Link-Kern 44 zu multiplexen und danach Paket für Paket auf die Busstruktur 58. Diese Informationen werden durch die übertragende Anwendung in einer Prioritätsmenge zu der Busstruktur geleitet. Der Demultiplexer 42 verwendet den Wert in dem Transaktionscode und die Transaktionsetikettfelder jedes von der Busstruktur 58 empfangenen Pakets und den Wert in dem Transaktionsetikett des Headers des asynchronen Antwortpakets dazu, das Paket zu dem entsprechenden asynchronen Datenübertragungskanal 20, 22 oder 24 zu leiten.

Bei dem asynchronen Datenübertragungskanal der vorliegenden Erfindung handelt es sich um einen bidirektionalen Datenpfad zwischen einem entsprechenden FIFO und dem Link-Kern 44. Be3i der Übertragung von Daten von dem entsprechenden FIFO zu dem Link-Kern 44 bildet der asynchrone Datenübertragungskanal die entsprechenden Header-Informationen und überträgt diese auf die Daten, bevor die resultierenden Header- und Anwendungsdaten zu dem Link-Kern 44 gesendet werden. Der Link-Block verwendet die durch den asynchronen Datenübertragungskanal erzeugten Daten zum Erzeugen und Abschließen der Schreibeoperation über die Busstruktur 58. Beim Übermitteln der Daten von dem Link-Kern 44 zu einem FIFO erzeugt der asynchrone Datenübertragungskanal die

BEST AVAILABLE COPY

entsprechenden Header-Informationen für die Lese-Transaktion. Der asynchrone Datenübertragungskanal übermittelt diese Informationen zu dem Link-Kern 44, der die Leseanforderung danach über die Busstruktur 58 übermittelt. Zu einem späteren Zeitpunkt führt der Antwortknoten ein Lese-Antwortpaket zurück. Der Link-Kern 44 erfasst dieses Antwortpaket und übermittelt dieses zu dem Demultiplexer 42, der die Daten daraufhin zu dem asynchronen Datenübertragungskanal leitet, der die Leseanforderung erzeugt hat, und zwar unter Verwendung der Werte in dem Transaktionscode und den Transaktionsetikettfeldern zur Ermittlung des entsprechenden asynchronen Datenübertragungskanals. Danach entfernt der asynchrone Datenübertragungskanal die Header-Informationen aus dem Paket und übermittelt die Daten zu dem entsprechenden FIFO. Die Anwendung verarbeitet danach die Daten von dem FIFO. Bei der Erzeugung von Lese- oder Schreib-Anforderungen, die über die Busstruktur 58 übermittelt werden sollen, erzeugt der asynchrone Datenübertragungskanal weiterhin die entsprechenden Anforderungen, bis der Kanal alle Daten zu oder von der Anwendung transportiert hat.

Ein System mit mehreren asynchronen Datenübertragungskanälen kann mehrere Threads bzw. Ablaufstränge von Datenübertragungen gleichzeitig erhalten. Dies ist bei eingebetteten Anwendungen nützlich, wie etwa bei Disketten- bzw. CD-Laufwerken, bei denen Mediendaten übertragen werden, während folgende Befehle gelesen oder Zustandsdaten an die einleitende Anwendung gesendet werden. Der Demultiplexer 42 ist dafür verantwortlich, die Daten entsprechend zu jedem asynchronen Datenübertragungskanal zu leiten. In dem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung weist jeder asynchrone Datenübertragungskanal ein eindeutiges Transaktionsetikett oder einen eindeutigen Bereich von

BEST AVAILABLE COPY

Transaktionsetiketten auf. Der Demultiplexer 42 bestimmt den entsprechenden asynchronen Datenübertragungskanal gemäß den Daten in dem Transaktionsetikett und den Transaktionscodefeldern.

Jeder asynchrone Datenübertragungskanal weist eine dedizierte Registerdatei auf, wie dies nachstehend im Text näher beschrieben wird. Die Registerdatei wird durch eine externe Intelligenz programmiert, wie etwa durch die Anwendung, aus der die Datenübertragungsoperation stammt. Sobald die Registerdatei programmiert ist, kann ein asynchroner Datenübertragungskanal Lese- und Schreib-Transaktionen entweder für einen größer werdenden Adressbereich oder eine feste Adresse über die Busstruktur 58 ausführen. Die Transaktionen können im Block- oder Quadlet-Format erfolgen. Bei der Programmierung der Datenübertragungsoperation liefert die Anwendung entweder einen Gesamtblockzählwert für die Übertragung, "erhöht" den Blockzähler jedes Mal um einen Zählwert oder sieht eine Kombination aus den beiden Möglichkeiten vor. Wenn ein Gesamtblockzählwert für die Übertragung programmiert wird, erzeugt der asynchrone Datenübertragungskanal die erforderlichen Transaktionen zum Abschluss der Operation, während die Anwendung andere Operationen ausführt und andere Aufgaben abschließt. Jeder asynchrone Datenübertragungskanal erhält den für die Busstruktur spezifischen Adresszählerkontext aufrecht und führt Lese- oder Schreib-Transaktionen durch, wenn der Blockzähler einen Wert von ungleich Null aufweist.

Jeder asynchrone Datenübertragungskanal erfordert eine dedizierte Registerdatei, die von der Ursprungsanwendung programmiert und dazu verwendet wird, die entsprechenden Transaktionen zu erzeugen, die für den Abschluss einer

BEST AVAILABLE COPY

Datenübertragungsoperation über die Busstruktur 58 erforderlich sind. Die für jeden asynchronen Datenübertragungskanal erforderliche Registerdatei, die Bestandteil des bevorzugten Ausführungsbeispiels der Erfindung ist, ist in der Abbildung aus Figur 3 veranschaulicht. Die Registerdatei 80 weist 32 Datenbytes auf, die hexadezimal von 0 bis 1F nummeriert sind. In der Abbildung aus Figur 3 ist die Registerdatei 80 in einem Tabellenformat mit acht horizontalen Zeilen dargestellt, die jeweils vier Bytes aufweisen. Die Abbildung aus Figur 3 weist eine Versatz-Spalte 82 auf, die den Versatz des Anfangsbytes in jeder Zeile von der Adresse es Anfangs der Registerdatei 80 darstellt. Eine Lese-/Schreib-Spalte 84 ist ebenfalls vorgesehen, um darzustellen, ob die Felder in jeder Zeile ausgelesen, beschrieben oder nur beschrieben werden können.

Bei dem Geschwindigkeitsfeld sp handelt es sich um ein Zwei-Bit-Feld in Byte 1 der Registerdatei 80. Das Geschwindigkeitsfeld sp kann ausgelesen und beschrieben werden. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem Geschwindigkeitsfeld sp. Eine Leseoperation aus dem Geschwindigkeitsfeld sp gibt den zuletzt in das Feld geschriebenen Wert wieder. Ein Wert in dem Geschwindigkeitsfeld stellt einen Zwei-Bit-Wert dar, der die Geschwindigkeit darstellt, mit der alle Anforderungspakete an der Busstruktur 58 erzeugt werden. Die folgende Tabelle definiert die Korrelation zwischen der Geschwindigkeit und dem Wert in dem Geschwindigkeitsfeld sp.

BEST AVAILABLE COPY

TABELLE I

Wert	Busgeschwindigkeit
00	100 Mbps
01	200 Mbps
10	400 Mbps
11	reserviert

Gemäß der Veranschaulichung in Tabelle I definiert ein Wert von 00 in dem Geschwindigkeitsfeld sp somit die Busgeschwindigkeit, mit der alle Anforderungspakete mit 100 Mbps erzeugt werden, wobei ein Wert von 01 einer Busgeschwindigkeit zur Erzeugung von Anforderungspaketen bei 200 Mbps entspricht, während ein Wert von 10 einer Busgeschwindigkeit zur Erzeugung von Anforderungspaketen mit 400 Mbps entspricht.

Das Transaktionsetikettfeld tl ist ein Sechs-Bit-Feld in Byte 2 der Registerdatei 80. Das Transaktionsetikettfeld tl kann ausgelesen und beschrieben werden. Das Transaktionsetikettfeld tl speichert den Wert des Transaktionsetiketts zur Verwendung für alle durch den entsprechenden asynchronen Datenübertragungskanal erzeugten Anforderungspakete. In einem alternativen Ausführungsbeispiel verwaltet ein einziger asynchroner Datenübertragungskanal einen Bereich von Transaktionsetiketten. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem Transaktionsetikettfeld tl. Eine Leseoperation aus dem Transaktionsetikettfeld tl gibt den zuletzt in das Feld geschriebenen Wert wieder. Wenn mehr als ein asynchroner Datenübertragungskanal in einem System vorgesehen sind, muss jeder asynchrone Datenübertragungskanal einen eindeutigen Wert in dem Transaktionsetikettfeld tl aufweisen, so dass der Demultiplexer 42 die Antwortpakete

BEST AVAILABLE COPY

entsprechend zu dem ursprünglichen asynchronen Datenübertragungskanal leiten.

In dem bevorzugten Ausführungsbeispiel werden die beiden wertniedrigsten Bits von Byte 2 der Registerdatei 80 dauerhaft auf einen logischen niedrigen Spannungswert programmiert.

Bei dem Transaktionscodefeld tCode handelt es sich um ein Vier-Bit-Feld in Byte 3 der Registerdatei 80. Das Transaktionscodefeld tCode kann ausgelesen und beschrieben werden. Das Transaktionscodefeld tCode speichert den Transaktionscode zur Verwendung für alle durch den entsprechenden asynchronen Datenübertragungskanal erzeugten Anforderungspakete. Eine Schreiboperation in das Feld aktualisiert den Wert in dem Transaktionscodefeld tCode. Eine Leseoperation aus dem Transaktionscodefeld tCode gibt den zuletzt in das Feld geschriebenen Wert wieder. Der Wert in dem Transaktionscodefeld tCode stellt einen Vier-Bit-Wert dar, der die Art der durchzuführenden Operation wiedergibt. Die Korrelation zwischen den Werten in dem Transaktionscodefeld tCode und der Art der durchzuführenden Operation wird in der nachstehenden Tabelle II veranschaulicht.

TABELLE II

tCode	Operation
0000	Schreibanforderung für Daten-Quadlet
0001	Schreibanforderung für Datenblock
0100	Leseanforderung für Daten-Quadlet
0101	Leseanforderung für Datenblock
1001	Sperroperation

BEST AVAILABLE COPY

Wenn das Transaktionscodefeld tCode den Wert 0000 aufweist, handelt es sich bei der auszuführenden Datenübertragungsoperation um eine Quadlet-Schreiboperation. Wenn das Transaktionscodefeld tCode den Wert 0001 aufweist, handelt es sich bei der auszuführenden Datenübertragungsoperation um eine Block-Schreiboperation. Wenn das Transaktionscodefeld tCode den Wert 0100 aufweist, handelt es sich bei der auszuführenden Datenübertragungsoperation um eine Quadlet-Leseoperation. Wenn das Transaktionscodefeld tCode den Wert 0101 aufweist, handelt es sich bei der auszuführenden Datenübertragungsoperation um eine Block-Leseoperation. Wenn das Transaktionscodefeld tCode den Wert 1001 aufweist, handelt es sich bei der Operation um eine Sperroperation.

In dem bevorzugten Ausführungsbeispiel werden die vier wertniedrigsten Bits von Byte 3 der Registerdatei 80 alle permanent auf einen logischen niedrigen Spannungswert programmiert, so dass ein reserviertes Feld in dem Paket-Header für die Busstruktur vorgesehen wird.

Bei dem Zielbezeichnerfeld destination_ID handelt es sich um ein Sechzehn-Bit-Feld in den Bytes 4 und 5 der Registerdatei 80. Das Zielbezeichnerfeld destination_ID kann ausgelesen und beschrieben werden. Das Zielbezeichnerfeld destination_ID speichert den Sechzehn-Bit-Zielknoten ID, der bei allen Anforderungspaketen verwendet wird, die von dem entsprechenden asynchronen Datenübertragungskanal für eine Datenübertragungsoperation erzeugt werden. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem Zielbezeichnerfeld destination_ID. Eine Leseoperation aus dem Zielbezeichnerfeld destination_ID gibt den zuletzt in das Feld geschriebenen Wert wieder. Der Wert in dem Zielbezeichnerfeld

BEST AVAILABLE COPY

destination_ID stellt den Knoten an der Busstruktur 58 dar, mit dem die Datenübertragungsoperation erfolgt. Jeder Knoten an der Busstruktur 58 weist somit einen eindeutigen Zielbezeichner auf.

Das höherwertige Zielversatzfeld destination_offset Hi ist ein Sechzehn-Bit-Feld in den Bytes 6 und 7 der Registerdatei 80. Das höherwertige Zielversatzfeld destination_offset Hi kann ausgelesen und beschrieben werden. Das höherwertige Zielversatzfeld destination_offset Hi speichert die höherwertigen sechszehn Bits der Zielversatzadresse zur Verwendung des nächsten erzeugten Anforderungspakets. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem höherwertigen Zielversatzfeld destination_offset Hi. Eine Leseoperation aus dem höherwertigen Zielversatzfeld destination_offset Hi gibt den aktuellen Wert der höherwertigen sechzehn Bits der Zielversatzadresse wieder.

Das niederwertige Zielversatzfeld destination_offset Lo ist ein Zweiunddreißig-Bit-Feld in den Bytes 8 bis B der Registerdatei. Das niederwertige Zielversatzfeld destination_offset Lo kann ausgelesen und beschrieben werden. Das niederwertige Zielversatzfeld destination_offset Lo speichert die niederwertigen zweiunddreißig Bits der Zielversatzadresse zur Verwendung für das nächste erzeugte Anforderungspaket. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem niederwertigen Zielversatzfeld destination_offset Lo. Eine Leseoperation aus dem niederwertigen Zielversatzfeld destination_offset Lo gibt den aktuellen Wert der niederwertigen zweiunddreißig Bits der Zielversatzadresse wieder. Gemeinsam bilden das höherwertige Zielversatzfeld destination_offset Hi und das niederwertige Zielversatzfeld destination_offset Lo die Achtundvierzig-Bit-

BEST AVAILABLE COPY

Zielversatzadresse, für die eine aktuelle Transaktion erzeugt wird. Wenn die Nicht-Inkrementierungs-Flagge in dem Steuerfeld, was nachstehend im Text näher beschrieben wird, einen logischen Niederspannungswert aufweist, inkrementiert der asynchrone Datenübertragungskanal das gesamte Achtundvierzig-Bit-Zielversatzfeld, welches das höherwertige Zielversatzfeld destination_offset Hi und das niederwertige Zielversatzfeld destination_offset Lo umfasst, um den Wert in dem Datenlängenfeld nach der Erzeugung jeder Lese- oder Schreibtransaktion.

Bei dem Datenlängenfeld data_length handelt es sich um ein Sechzehn-Bit-Feld in den Bytes C und D der Registerdatei 80. Das Datenlängenfeld data_length kann ausgelesen und beschrieben werden. Das Datenlängenfeld data_length speichert in Bytes die Größe aller Anforderungspakete, die durch den entsprechenden asynchronen Datenübertragungskanal erzeugt werden. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem Datenlängenfeld data_length. Eine Leseoperation aus dem Datenlängenfeld data_length gibt den letzten in das Feld geschriebenen Wert wieder. Der Wert in dem Datenlängenfeld data_length unterliegt einigen Einschränkungen, und zwar auf der Basis der Werte in den anderen Feldern der Registerdatei, wie dies in der folgenden Tabelle III dargestellt ist.

BEST AVAILABLE COPY

TABELLE III

Operation	Tcode	Erweiterter tCode	sp	Zulässiger Wert f. data_length (Bytes)
Quadlet-Lesen/Quadlet-Schreiben	0100/0000	0000	-	4
Block-Lesen/Block-Schreiben	0101/0001	0000	00	1 bis 512
Block-Lesen/Block-Schreiben	0101/0001	0000	01	1 bis 1024
Block-Lesen/Block-Schreiben	0101/0001	0000	10	1 bis 2048
mask_swap	1001	0001	-	8 oder 16
compare_swap	1001	0002	-	8 oder 16
fetch_add	1001	0003	-	4 oder 8
little_add	1001	0004	-	4 oder 8
bounded_add	1001	0005	-	8 oder 16
wrap_add	1001	0006	-	8 oder 16
Anbieter-abhängig	1001	0007	-	-

Das erweiterte Transaktionsfeld extended_tCode ist ein Sechzehn-Bit-Feld in den Bytes E und F der Registerdatei 80. Das erweiterte Transaktionsfeld extended_tCode kann ausgelesen und beschrieben werden. Eine Schreiboperation in dieses Feld aktualisiert den Wert in dem erweiterten Transaktionsfeld extended_tCode. Eine Leseoperation aus dem erweiterten Transaktionsfeld extended_tCode gibt den zuletzt in dieses Feld geschriebenen Wert wieder. Das erweiterte Transaktionsfeld extended_tCode weit für alle Transaktionen mit Ausnahme der Sperr-Transaktionen einen Wert von Null auf.

BEST AVAILABLE COPY

Wenn der Wert in dem Transaktionsfeld tCode auf einen Wert von 1001 gesetzt wird, wodurch angegeben wird, dass es sich dabei um eine Sperranforderung handelt, so speichert das erweiterte Transaktionsfeld extended_tCode den erweiterten Transaktionscodewert für die Sperr-Transaktion.

Bei dem Paketzählerfeld handelt es sich abhängig von der Konfiguration des Systems um ein Acht- bis Zweiunddreißig-Bit-Feld in den Bytes 10 bis 13 der Registerdatei 80. Das Paketzählerfeld kann ausgelesen und beschrieben werden. Das Paketzählerfeld speichert die Anzahl der zur Beendigung der Datenübertragungsoperation verbleibenden zu erzeugenden Anforderungspakete. Eine Schreiboperation in dieses Feld ändert den Wert in dem Paketzählerfeld. Eine Leseoperation aus dem Paketzählerfeld gibt den aktuellen Paketzählwert der verbleibenden zu erzeugenden Anforderungspakete wieder. Der Wert in dem Paketzählerfeld wird nach der Erzeugung jeder Transaktion herabgesetzt. Zur Gewährleistung einer umfassenden Kontrolle über die Anzahl der erzeugten Pakete darf das Paketzählerfeld nur dann beschrieben werden, wenn der Wert des Feldes gleich Null ist.

Bei dem Paketzähler-Erhöhungsfeld handelt es sich um ein lesegeschütztes Feld in den Bytes 14 bis 17 der Registerdatei. Wenn das Paketzähler-Erhöhungsfeld beschrieben wird, erhöht der entsprechende asynchrone Datenübertragungskanal den Wert in dem Paketzählerregister. Wenn das Paketzähler-Erhöhungsfeld ausgelesen wird, ist der zurückgegebene Wert nicht vorhersehbar. Dies ermöglicht es, dass die Ursprungsanwendung zusätzliche Transaktionen für eine aktuelle Datenübertragungstransaktion erzeugt. In dem bevorzugten Ausführungsbeispiel kann nur durch Beschreiben des Paketzähler-Erhöhungsfeldes der Wert in dem Paketzählerfeld

BEST AVAILABLE COPY

erhöht werden, wenn das Paketzählerfeld einen Wert von ungleich Null aufweist.

Bei dem Steuerfeld handelt es sich um ein Zweiunddreißig-Bit-Feld in den Bytes 18 bis 1B der Registerdatei 80. Das Steuerfeld kann ausgelesen und beschrieben werden. In dem Steuerfeld sind die Bits 0 bis 29 reserviert, wobei es sich bei dem Bit 30 um ein Nicht-Inkrementierungs-Steuerbit non_incr und bei dem Bit 31 um ein Betriebssteuerbit go handelt. Das Betriebssteuerbit go wird auf einen logischen hohen Spannungswert gesetzt, um den asynchronen Datenübertragungskanal freizugeben. Die Freigabe des Betriebssteuerbits go auf einen logischen niedrigen Spannungswert deaktiviert den asynchronen Datenübertragungskanal sofort oder an der nächsten Transaktionsgrenze, wenn sich der asynchrone Datenübertragungskanal gerade mitten in einer Transaktion befindet. Demgemäß ist ein asynchroner Datenübertragungskanal nur dann funktionsfähig, wenn das Betriebssteuerbit go auf einen logischen hohen Spannungswert gesetzt ist. Das Nicht-Inkrementierungs-Steuerbit non_incr wird auf einen logischen hohen Spannungswert gesetzt, um zu bewirken, dass der asynchrone Datenübertragungskanal alle Anforderungspakete für eine feste bzw. Nicht-Inkrementierungs-Adresse erzeugt. Wenn das Nicht-Inkrementierungs-Steuerbit non_incr einem logischen niedrigen Spannungswert entspricht, inkrementiert der entsprechende asynchrone Datenübertragungskanal den Zielversatzwert um den Wert in dem Feld data_length nach Beendigung jeder Transaktion.

Bei dem Zustandsfeld handelt es sich um ein Zweiunddreißig-Bit-Feld in den Bytes 1C bis 1F der Registerdatei 80. Das Zustandsfeld kann ausgelesen und beschrieben werden. Das

BEST AVAILABLE COPY

Zustandsfeld speichert die letzten Quittungscodes und Antwortcodes, die von den Anforderungspaketen resultieren, die durch den entsprechenden asynchronen Datenübertragungskanal erzeugt werden. Das Zustandsfeld weist ein Fehlerfeld, ein Antwortcodefeld, ein Quittung-Ein-Feld und ein Quittung-Aus-Feld auf.

Bei dem Fehlerfeld handelt es sich um ein Vier-Bit-Feld, das Bits aufweist, die den Fehler anzeigen, der das Anhalten des Betriebs des entsprechenden asynchronen Datenübertragungskanals bewirkt hat. Das Fehlerfeld wird gelöscht, wenn das Betriebssteuerbit go auf einen logischen hohen Spannungswert gesetzt wird. Das Fehlerfeld ist gültig, wenn das Betriebssteuerbit go durch den asynchronen Datenübertragungskanal auf einen logischen niedrigen Spannungswert gelöscht wird. Die Tabelle IV veranschaulicht das Verhältnis zwischen den möglichen Werten in dem Fehlerfeld und deren Bedeutung.

TABELLE IV

Fehlerwert	Bedeutung
0000	Kein Fehler
0001	ungült. Quittungscode empf. (für Anforderungspaket)
0010	ungült. Quittungscode gesendet (für Antwortpaket)
0100	Geteilte Transaktion Timeout
1000	Bus-Rücksetzung eingetreten

Ein Wert von 0000 in dem Fehlerfeld zeigt an, dass kein Fehler vorliegt. Ein Wert von 0001 in dem Fehlerfeld zeigt an, dass

BEST AVAILABLE COPY

der Fehler dadurch bewirkt worden ist, dass ein ungültiger Quittungscode für ein vorher übermitteltes Anforderungspaket empfangen worden ist. Ein Wert von 0010 in dem Fehlerfeld zeigt an, dass der Fehler durch das Senden eines ungültigen Quittungscodes für ein Antwortpaket bewirkt worden ist. Ein Wert von 0100 in dem Fehlerfeld zeigt an, dass der Fehler durch ein auftretendes Timeout für eine geteilte Transaktion bewirkt worden ist. Ein Wert von 1000 in dem Fehlerfeld zeigt an, dass eine Rücksetzung des Busses eingetreten ist.

Das Antwortcodefeld rcode ist ein Vier-Bit-Feld, das den letzten empfangenen Antwortcode speichert. Der Wert in dem Antwortcodefeld ist gleich 1111, wenn es sich bei der letzten Transaktion um eine Schreibtransaktion gehandelt hat, die als unifizierte Transaktion ausgeführt worden ist.

Das Quittung-Ein-Feld ist ein Vier-Bit-Feld, das das letzte empfangene Quittungssignal von dem entfernten Knoten als Reaktion auf das letzte durch den asynchronen Datenübertragungskanal empfangene Anforderungspaket speichert.

Das Quittung-Aus-Feld ist ein Vier-Bit-Feld, das das letzte durch den asynchronen Datenübertragungskanal erzeugte Quittungssignal speichert, und zwar als Reaktion auf ein Antwortpaket, das einem durch den entsprechenden asynchronen Datenübertragungskanal erzeugten Anforderungspaket entspricht.

Eine Schreiboperation in das Zustandsfeld ändert den Wert in dem Feld. Eine Leseoperation aus dem Feld gibt den aktuellen Zustand des asynchronen Datenübertragungskanals und der aktuellen Datenübertragungsoperation wieder. Wenn eines der Anforderungspakete oder ein entsprechendes Antwortpaket zu einem Fehler führt, erzeugt der asynchrone

BEST AVAILABLE COPY

Datenübertragungskanal fürs erste keine weiteren Anforderungspakete. Danach speichert der asynchrone Datenübertragungskanal die Werte für das Antwortcodefeld rcode, das Quittung-Ein-Feld ack-in und das Quittung-Aus-Feld ack-out in dem Zustandsfeld. Nach dem Speichern dieser Werte in dem Zustandsfeld sieht der asynchrone Datenübertragungskanal ein Interrupt-Signal über die Anwendungsschnittstelle bzw. die Programmierschnittstelle für die Anwendung vor, um der Anwendung mitzuteilen, dass ein Fehlerzustand während der aktuellen Datenübertragungsoperation aufgetreten ist.

Leseoperationen

Beim Ausführen einer Leseoperation und der Gewinnung von Daten von einem anderen mit der Busstruktur verbundenen Knoten sowie der Übertragung von Daten zu der Anwendung, erzeugt ein asynchroner Datenübertragungskanal die entsprechenden Lese-Anforderungspakete unter Verwendung der Informationen in der Registerdatei 80 als Mustervorlage. Wenn die Daten danach von dem Zielknoten empfangen werden, leitet der Demultiplexer 42 die Daten zu dem entsprechenden asynchronen Datenübertragungskanal unter Verwendung der Werte in den Transaktionscode- und Transaktionsetikettfeldern. Der asynchrone Datenübertragungskanal entfernt daraufhin die Header-Informationen aus den Datenpaketen und lädt die Datenpakete in das FIFO, wobei die Anwendung von dort die empfangenen Daten verarbeiten kann.

Im aktiven Zustand und bei der Übertragung von Daten von der Busstruktur 58 zu der FIFO-Schnittstelle arbeitet jeder asynchrone Datenübertragungskanal als Datenempfangs-

BEST AVAILABLE COPY

Zustandsvorrichtung gemäß der Definition in der folgenden Tabelle V.

TABELLE V

```
while (Active ()) {
     if (RAM_Data==0)          /*wenn keine Daten zu entladen*/
          continue;            /*loop um akt.Zust.z.prüfen*/
                               /*freier Raum u. aktiv*/
     AssertReq();              /*assert Anford.*/
     While(!Ack()              /*auf Quitt.warten*/
          &&Active());         /*sicherstellen, dass weiter akt.*/

     if (!Active())            /*verlassen, wenn nicht mehr akt.*/
          break;

     AssertWord();             /*Wort an FIFO-Schnittst.ass.*/
     DeAssertReq();            /*Deass. Anford.*/
}
```

Die FIFO-Schnittstelle taktet die Daten von dem asynchronen Datenübertragungskanal in das entsprechende FIFO mit einem Taktsignal, das mit der Busstruktur-Schnittstelle synchronisiert ist. Das FIFO weist immer einen Zustand für den Empfang eines Datenwortes auf, wenn dieses von dem asynchronen Datenübertragungskanal zur Verfügung steht. Wenn das Anforderungssignal geltend gemacht wird und kein Platz in dem FIFO vorhanden ist, kommt es zu einem FIFO-Überlauf. Dies erzeugt einen Fehlerzustand, der durch den entsprechenden asynchronen Datenübertragungskanal erfasst wird. Wenn ein FIFO-Überlauf eintritt, werden die verbleibenden Transaktionen angehalten, bis das FIFO gelöscht und für den Empfang

BEST AVAILABLE COPY

zusätzlicher Daten bereit ist. In diesem Fall reflektiert das Quittung-Aus-Feld des Zustandsregisters den Fehler.

Zum Lesen von Daten von der Busstruktur programmiert die Ursprungsanwendung die entsprechenden Informationen in die Registerdatei für den entsprechenden asynchronen Datenübertragungskanal. Der entsprechende Wert für die zu verwendete Busgeschwindigkeit, entweder 100 Mbps, 200 Mbps oder 400 Mbps, wird in das Geschwindigkeitsfeld sp programmiert. Die zu verwendende Busgeschwindigkeit muss innerhalb des Funktionsbereichs der physikalischen Schnittstelle 56 liegen und von der Busstruktur 58 unterstützt werden. Der entsprechende Wert für die spezifische, abzuschließende Transaktion wird in das Transaktionscodefeld tCode programmiert. Der entsprechende Wert, der dem Bezeichner des Zielknotens an der Busstruktur für alle Anforderungspakete entspricht, wird in das Zielbezeichnerfeld destination_ID programmiert.

Der anfängliche Achtundvierzig-Bit-Zielversatzwert wird in die hohen und niedrigen Zielversatzfelder destination_offset Hi und destination_offset Lo programmiert. Wenn das Nicht-Inkrementierungs-Bit in dem Steuerfeld einen logischen niedrigen Spannungswert aufweist, wird der Wert in den Zielversatzfeldern nach jeder erzeugten Anforderungstransaktion inkrementiert. Die Anzahl der Bytes für jedes zu erzeugende Anforderungspaket wird in das Datenlängenfeld data_length programmiert. Wenn der Wert in dem Transaktionscodefeld tCode gleich 0100 ist, was anzeigt, dass es sich bei der Transaktion um eine Quadlet-Lesetransaktion handelt, so ist der Wert in dem Datenlängenfeld data_length gleich Vier. Wenn der Wert in dem Transaktionscodefeld tCode gleich 0101 ist, was anzeigt, dass diese Transaktion eine

BEST AVAILABLE COPY

Block-Lesetransaktion darstellt, so wird der Wert in dem Datenlängenfeld data_length mit einem entsprechenden Wert in dem Bereich der zulässigen Zahlen für die programmierte Busgeschwindigkeit programmiert, wie dies in der vorstehenden Tabelle III dargestellt ist. Da es sich bei der durchzuführenden Operation um eine Leseoperation und nicht um eine Sperr-Transaktion handelt, wird der Wert in dem erweiterten Transaktionscodefeld extended_tCode als gleich Null programmiert.

Die Anzahl der zu erzeugenden und zu übermittelnden Pakete zum Abschluss dieser Datenübertragungsoperation wird in das Paketzählerfeld programmiert. Der Wert in dem Paketzählerfeld kann anfangs mit gleich Null programmiert werden, wenn die Anwendung in das Paketzähler-Erhöhungsfeld schreibt, um die entsprechenden Transaktionen nacheinander zu erzeugen. Das Nicht-Inkrementierungs-Bit in dem Steuerfeld wird so programmiert, dass es gleich einem logischen hohen Spannungswert ist, wenn alle Anforderungspakete an die gleiche Zielversatzadresse übermittelt werden sollen. Das Nicht-Inkrementierungs-Bit in dem Steuerfeld wird auf einen Wert gleich dem logischen niedrigen Spannungswert programmiert, wenn die Anforderungspakete an einen ansteigenden Adressbereich übermittelt werden sollen. Das Betriebssteuerbit go in dem Steuerfeld wird auf einen Wert gleich einem logischen hohen Spannungswert programmiert, um es zu ermöglichen, dass der asynchrone Datenübertragungskanal mit der Erzeugung der entsprechenden Transaktionen beginnt, die erforderlich sind, um die Datenübertragungsoperation abzuschließen.

Wenn das Betriebssteuerbit go in dem Steuerfeld auf einen logischen hohen Spannungswert gesetzt wird, tritt der

BEST AVAILABLE COPY

asynchrone Datenübertragungskanal in den aktiven Zustand ein. Im aktiven Zustand erzeugt der asynchrone Datenübertragungskanal Lese-Anforderungspakete gemäß der Lese-Zustandsvorrichtung gemäß der Definition in Tabelle VI.

TABELLE VI

```
while(Active ()) {
      if(packet_counter==0)                  /*wenn keine Pakete z. send.*/
            continue;                        /*loop, um akt. Zust.z.verif.*/

      if((RAM_Free<data_length)              /*wenn nicht genug freier Platz*/
        &&(RAM_Data!=0==                     /*und n. nicht leer*/
            continue;                        /*loop, um akt. Zust.z.verify.*/

                                             /*genug Raum für ein Paket*/

      Arbitrate ();                          /*Zugriff auf Link-Kern*/
      if(tCode==4)                           /*Wenn Quadlet*/
          SendHeaderRegs(12);                /*ersten 12 Bytes an Headerr. s.*/
      Else                                   /*ist Block*/
            SendHeaderRegs(16);              /*erst.16 Bytes a.Headerr. send.*/

      /*Hier müssen ungült. Quittungen bearbeitet werden*/

      GetData (data_length,                  /*empf.Daten in Puffer-RAM*/
        &RAM_Data, &RAM_Free);               /*als angek.Daten anpass.*/

      /*Hier müssen rcodes, die zurückkommen, bearb. werden*/

      --packet_counter;                      /*Paketzähler dekrem.*/
      if(!non_increment)                     /*falls inkrementierend*/
        destination_offset+=data_length;     /*Ziel inkrementieren*/
}
```

Die Ursprungsanwendung kann jederzeit in das Paketzähler-Erhöhungsfeld schreiben, wodurch der Wert in dem

BEST AVAILABLE COPY

Paketzählerfeld um Eins erhöht wird. Die Lesezustandsvorrichtung des asynchronen Datenübertragungskanals erzeugt gemäß der Definition in der vorstehenden Tabelle VI immer dann ein Lese-Anforderungspaket, wenn in dem mit dem aktiven asynchronen Datenübertragungskanal ein größerer freier Raum als für ein Paket vorhanden ist. Die Lesezustandsvorrichtung des asynchronen Datenübertragungskanals erzeugt auch dann ein Leseanforderungspaket, wenn das dem asynchronen Datenübertragungskanal entsprechende FIFO völlig leer ist. Wenn die eingebettete Anwendung gewährleistet, dass die Daten schnell genug aus dem entsprechenden FIFO getaktet werden und mit einer ausreichend kurzen Latenz, so kann die Größe des dem asynchronen Datenübertragungskanal entsprechenden FIFO kleiner sein als die Anzahl der Bytes, die durch den Wert in dem Datenlängenfeld data_length in der Registerdatei 80 spezifiziert sind.

Für jedes Lese-Anforderungspaket, das durch den asynchronen Datenübertragungskanal erzeugt wird, erwartet der asynchrone Datenübertragungskanal, dass der Zielknoten ein entsprechendes Lese-Antwortpaket erzeugt. Der Demultiplexer verwendet den Transaktionscode tCode und das Transaktionsetikett tl in dem Lese-Antwortpaket, um das Paket zu dem entsprechenden asynchronen Datenübertragungskanal zu leiten, wenn in einem System mehrere asynchrone Datenübertragungskanäle vorgesehen sind. Der empfangende asynchrone Datenübertragungskanal entfernt dann den Header und macht das Datenfeld an der entsprechenden FIFO-Schnittstelle verfügbar.

Wenn nach der Erzeugung jedes Lese-Anforderungspakets das Nicht-Inkrementierungs-Bit in dem Steuerfeld nicht auf einen logischen hohen Spannungswert gesetzt ist, erhöht der

BEST AVAILABLE COPY

asynchrone Datenübertragungskanal den Zielversatzadressenwert um den Wert in dem Datenlängenfeld data_length zur Vorbereitung auf die Erzeugung des nächsten Lese-Anforderungspakets. Obwohl dies in der Definition der Lese-Zustandsvorrichtung aus der vorstehenden Tabelle VI nicht vorgesehen ist, untersucht der asynchrone Datenübertragungskanal die Quittung in dem Feld für jedes erzeugte Schreib-Anforderungspaket und das Antwortcodefeld rcode für jedes entsprechende Lese-Antwortpaket. Wenn entweder die Quittung in dem Feld oder das Antwortcodefeld rcode einen Fehler anzeigt, oder wenn der asynchrone Datenübertragungskanal einen ungültigen Quittungscode für das Lese-Antwortpaket aufgrund eines Fehlers zurückführen muss, hält der asynchrone Datenübertragungskanal sofort an und speichert sowohl die Quittungscodes als auch den Antwortcode rcode in dem Zustandsfeld des asynchronen Datenübertragungskanals in der Registerdatei 80. Bei geteilten Transaktionen misst der asynchrone Datenübertragungskanal die Zeit der Antwort. Wenn zwischen dem Anforderungspaket und dem entsprechenden Antwortpaket mehr als 100 Millisekunden liegen, stoppt der asynchrone Datenübertragungskanal und zeigt die definierten Zustandsinformationen in dem Zustandsfeld der Registerdatei 80 an.

Schreiboperationen

Bei der Ausführung einer Schreiboperation und dem Senden von Daten von der Ursprungsanwendung an einen anderen Knoten, der mit der Busstruktur gekoppelt ist, erzeugt ein asynchroner Datenübertragungskanal einen entsprechenden Header unter Verwendung der Informationen in der Registerdatei 80 als Mustervorlage. Der Header wird danach einem entsprechenden Datenpaket hinzugefügt, und sowohl der Header als auch das

BEST AVAILABLE COPY

Datenpaket werden durch den Link-Kern 44 auf die Busstruktur 58 übertragen. Wenn die Inkrementierungsfunktion nicht deaktiviert ist, erhöht der asynchrone Datenübertragungskanal den Wert in den Zielversatzfeldern und erzeugt den Header für das nächste Datenpaket. Nach der Erzeugung jeder Transaktion wird der Paketzählerwert herabgesetzt. Dieser Prozess wird solange wiederholt, bis der Wert in dem Paketzählerfeld gleich Null ist.

Wenn ein asynchroner Datenübertragungskanal aktiv ist und Daten von dem FIFO zu der Busstruktur 58 überträgt, so fungiert der Kanal als eine Datenübermittlungs-Zustandsvorrichtung gemäß der nachstehenden Definition in Tabelle VII.

TABELLE VII

```
while (Active ()) {
      if (RAM-Free==0)          /*ohne freiem Platz*/
          continue;             /*loop, zum Prüfen v. akt.Zustand*/


      AssertReq ();             /*freier Platz und aktiv*/
      while (!Ack()             /*Assert Anford.*/
            &&Active());        /*Auf Quittung warten*/

      if(!Active())             /*verlassen, wenn n. mehr aktiv*/
            break;

      LatchWord();              /*Wort speichern*/
      DeAssertReq();            /*Deassert Anford.*/
}
```

Die FIFO-Schnittstelle taktet die Daten von dem FIFO zu dem entsprechenden asynchronen Datenübertragungskanal mit einem

BEST AVAILABLE COPY

Takt, der mit der Schnittstelle der Busstruktur synchronisiert ist. Die FIFO weist immer ein verfügbares Datenwort auf, wenn der asynchrone Datenübertragungskanal ein Wort anfordert. Wenn das Anforderungssignal Req geltend gemacht wird, wenn keine Daten in dem FIFO sind, so kommt es zu einem FIFO-Unterlauf. Dies erzeugt einen Fehler, der von dem entsprechenden asynchronen Datenübertragungskanal erfasst und bearbeitet wird. Die Anwendung ist dafür verantwortlich zu gewährleisten, dass die entsprechenden Daten in dem FIFO zur Übertragung über die Busstruktur 58 gespeichert werden. Wenn ein FIFO-Unterlauf eintritt, werden die verbleibenden Transaktionen angehalten, bis das FIFO zusätzliche zu übermittelnde Daten aufweist.

Zum Schreiben von Daten auf die Busstruktur 58 programmiert die Anwendung die entsprechenden Informationen in die Registerdatei für den entsprechenden asynchronen Datenübertragungskanal. Der entsprechende zu verwendende Wert für die Busgeschwindigkeit, 100 Mbps, 200 Mbps oder 400 Mbps, wird in das Geschwindigkeitsfeld sp programmiert. Die zu verwendende Busgeschwindigkeit wird so ausgewählt, dass sie innerhalb des Funktionsbereichs der physikalischen Schnittstelle 56 liegt und von der Busstruktur 58 unterstützt wird. Der entsprechende Wert für die spezifische Transaktion, die abzuschließen ist, wird in das Transaktionscodefeld tCode programmiert. Wenn es sich bei den Anforderungen um Quadlet-Schreibanforderungen handelt, wird ein Wert von 0000 in das Transaktionscodefeld tCode programmiert. Wenn es sich bei den Anforderungen um Block-Schreibanforderungen handelt, wird ein Wert von 0001 in das Transaktionscodefeld tCode programmiert. Der entsprechende Wert, der dem Bezeichner des Zielknotens an der Busstruktur für alle Anforderungspakete entspricht, wird in das Zielbezeichnerfeld destination_ID programmiert.

BEST AVAILABLE COPY

Der anfängliche Achtundvierzig-Bit-Zielversatzwert wird in die hohen und niedrigen Zielversatzfelder destination_offset Hi und destination_offset Lo programmiert. Wenn das Nicht-Inkrementierungs-Bit in dem Steuerregister einen logischen niedrigen Spannungswert aufweist, wird der Wert in den Zielversatzfeldern der Registerdatei 80 nach Abschluss jeder Anforderungstransaktion erhöht. Die Anzahl der Bytes für jedes zu erzeugende Anforderungspaket wird in das Datenlängenfeld data_length programmiert. Wenn der Wert in dem Transaktionscodefeld tCode gleich 0000 ist, was anzeigt, dass diese Transaktion eine Quadlet-Schreibtransaktion darstellt, so ist der Wert in dem Datenlängenfeld data_length gleich Vier. Wenn der Wert in dem Transaktionscodefeld tCode gleich 0001 ist, was anzeigt, dass es sich bei dieser Transaktion um eine Block-Schreibtransaktion handelt, so wird der Wert in dem Datenlängenfeld data_length mit einem entsprechenden Wert im Bereich der zulässigen Zahlen für die programmierte Busgeschwindigkeit programmiert, wie dies in der vorstehenden Tabelle III dargestellt ist. Da es sich bei der durchgeführten Transaktion um eine Schreiboperation handelt, wird der Wert in dem erweiterten Transaktionscodefeld extended_tCode mit gleich Null programmiert.

Die Anzahl der zu erzeugenden Pakete, die zu übermitteln sind, um diese Transaktion abzuschließen, wird in das Paketzählerfeld programmiert. Der Wert in dem Paketzählerfeld kann anfänglich mit gleich Null programmiert werden, wenn die Anwendung das Paketzähler-Erhöhungsfeld beschreibt, um nacheinander die entsprechenden Transaktionen zu erzeugen. Das Nicht-Inkrementierungs-Bit in dem Steuerfeld wird so programmiert, dass es dem logischen hohen Spannungswert entspricht, wenn alle Anforderungspakete an die gleiche Zielversatzadresse gesendet werden sollen. Das Nicht-

BEST AVAILABLE COPY

Inkrementierungs-Bit in dem Steuerfeld wird so programmiert, dass es einem logischen niedrigen Spannungswert entspricht, wenn die Anforderungspakete an einen ansteigenden Adressbereich übermittelt werden sollen. Das Betriebssteuerbit go in dem Steuerfeld wird so programmiert, dass es einem logischen hohen Spannungswert entspricht, so dass der asynchrone Datenübertragungskanal mit dem Erzeugen der entsprechenden Transaktionen beginnen kann, die erforderlich sind, um die Datenübertragungsoperation abzuschließen.

Wenn das Betriebssteuerbit go in dem Steuerfeld der Registerdatei 80 auf einen logischen hohen Spannungswert gesetzt wird, tritt der asynchrone Datenübertragungskanal in den aktiven Zustand ein. Im aktiven Zustand erzeugt der asynchrone Datenübertragungskanal Anforderungspakete gemäß der Schreib-Zustandsvorrichtung gemäß der Definition in der folgenden Tabelle VIII.

BEST AVAILABLE COPY

TABELLE VIII

```
while (Active()) {
      if(packet_counter==0)                  /*wenn k.Pak. zu senden*/
                  continue                   /*Loop, um ak.Z.z.verif.*/

      if(((RAM_Data<data_length)             /*wenn nicht gen. Daten*/
                  &&(RAM_Free!=0))           /*und n. nicht voll*/
                   continue;                 /*loop, um ak. Z. z ver.*/
                                             /*genug Daten f. ein Paket*/

      Arbitrate ();                          /*Zugr. Auf Link-Kern*/
      if(tCode==0)                           /*wenn Quadlet*/
                  SendHeaderRegs(12);        /*erste 12 Bytes v. Head.r.s.*/

      Else                                   /*ansonsten ist es Block*/
                  SendHeaderRegs(16);        /*erst.16 Bytes v. Head.r.se.*/

      SendData (data_length,                 /*Datenf.v.Puffer-RAM senden*/
        &RAM_Data, &RAM_Free);               /*bei Dat.übertr. anpassen*/
      if(ack==pending)                       /*bei anh.Quitt.code*/
        WaitResponse();                      /*auf Antwortpaket warten*/

/*Hinweis: hier müssen ung. Quitt.codes u. ung.rcodes bearb. werden*/

      --packet_counter;                      /*Paketzähler herabsetzen*/
      if(!non_increment)                     /*falls inkrementierend*/
        destination_offset+=data_length;     /*Ziel inkrementieren*/
}
```

Zu jedem Zeitpunkt kann die Ursprungsanwendung das Paketzähler-Erhöhungsfeld beschreiben und dadurch den Wert in dem Paketzählerfeld um Eins erhöhen. Die Schreib-Zustandsvorrichtung des asynchronen Datenübertragungskanals bildet gemäß der Definition in der obigen Tabelle VIII ein Schreib-Anforderungspaket, wenn mehr Daten als für ein Paket in dem FIFO vorgesehen sind, das mit dem aktiven asynchronen

BEST AVAILABLE COPY

Datenübertragungskanal verbunden ist. Die Schreib-Zustandsvorrichtung des asynchronen Datenübertragungskanals bildet ferner ein Schreib-Anforderungspaket, wenn das FIFO, das dem asynchronen Datenübertragungskanal entspricht, ganz gefüllt ist. Wenn die eingebettete Anwendung gewährleistet, dass die Daten in dem entsprechenden FIFO schnell genug gesperrt werden und mit ausreichend kurzer Latenz, kann die Größe des dem asynchronen Datenübertragungskanal entsprechenden FIFO kleiner sein als die Anzahl der Bytes, die durch den Wert in dem Datenlängenfeld data_length in der Registerdatei 80 spezifiziert ist.

Nach der Erzeugung jedes Schreib-Anforderungspakets, wenn das Nicht-Inkrementierungs-Bit in dem Steuerfeld nicht auf einen logischen hohen Spannungswert gesetzt ist, erhöht der asynchrone Datenübertragungskanal den Zielversatzadresswert um den Wert in dem Datenlängenfeld data_length in Vorbereitung auf die Erzeugung des nächsten Schreib-Anforderungspakets. Obwohl dies in der in Tabelle VIII definierten Schreib-Zustandsvorrichtung nicht dargestellt ist, untersucht der asynchrone Datenübertragungskanal die Quittung in dem Feld für jedes Schreib-Anforderungspaket, das erzeugt wird und dem Antwortcodefeld rcode, wenn er Zielknoten ein Schreib-Antwortpaket erzeugt. Wenn die Quittung in dem Feld oder das Antwortcodefeld rcode einen Fehler anzeigt oder wenn der asynchrone Datenübertragungskanal aufgrund eines Fehlers einen ungültigen Quittungscode für das Schreib-Antwortpaket zurückführen muss, hält der asynchrone Datenübertragungskanal sofort an und speichert sowohl die Quittungscodes als auch den Antwortcode rcode in dem Zustandsfeld des asynchronen Datenübertragungskanals in der Registerdatei 80. Bei geteilten Transaktionen stoppt der asynchrone Datenübertragungskanal die zeit der Antwort. Wenn zwischen dem Anforderungspaket und dem

BEST AVAILABLE COPY

entsprechenden Antwortpaket mehr als 100 Millisekunden liegen, hält der asynchrone Datenübertragungskanal an und zeigt die definierten Zustandsinformationen in dem Zustandsfeld der Registerdatei 80 an.

In dem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung handelt es sich bei der Busstruktur 58 um eine Busstruktur des IEEE 1394 Standards. Somit erzeugt jeder asynchrone Datenübertragungskanal Transaktionen, Header, Anforderungen und Antworten in dem von dem IEEE 1394 Standard vorausgesetzten Format. Für den Fachmann ist es ersichtlich, dass der erfindungsgemäße asynchrone Datenübertragungskanal in Verbindung mit anderen Arten von Busstrukturen und Systemen eingesetzt werden kann. Bei solchen Systemen wird der asynchrone Datenübertragungskanal so angepasst, dass er für die entsprechende Busstruktur jeweils die angemessenen Transaktionen, Header, Anforderungen und Antworten erzeugt.

Die vorliegende Erfindung wurde durch spezifische Ausführungsbeispiele mit Details beschrieben, um das Verständnis der Grundsätze der Konstruktion und der Funktionsweise der Erfindung zu erleichtern. Alle Bezugnahmen auf die spezifischen Ausführungsbeispiele und Details schränken den Umfang der anhängigen Ansprüche in keiner Weise ein.

BEST AVAILABLE COPY

Patentansprüche

1. Asynchroner Datenübertragungskanal (20), der so konfiguriert ist, daß er eine Verbindung zwischen einer Anwendung (12) und einer IEEE-1394-Busstruktur vorsieht, um asynchrone Datenübertragungsoperationen zu und von der Anwendung über die Busstruktur (58) zu verwalten, wobei der Datenübertragungskanal folgendes umfaßt:

(a) eine Programmierschaltung, die mit einer Registerdatei (26) verbunden und so konfiguriert ist, daß sie mit der Anwendung (12) derart verbunden ist, daß sie Anweisungen in Bezug auf eine Datenübertragungsoperation von der Anwendung (12) empfängt und die Anweisungen in der Registerdatei (26) speichert; und

(b) eine automatische Transaktionserzeugungsschaltung, die mit der Registerdatei (26) gekoppelt ist, um automatisch die für die Vollendung der Datenübertragungsoperation erforderlichen Transaktionen zu erzeugen, und zwar ohne unmittelbare Prozessorsteuerung oder Überwachung durch die Anwendung (12).

2. Asynchroner Datenübertragungskanal (20) nach Anspruch 1, wobei die Registerdatei (26) als Vorlage für die Erzeugung der Transaktionen und Header verwendet wird, die für die Vollendung der Datenübertragungsoperation erforderlich sind, und zwar ohne unmittelbare Prozessorsteuerung oder Überwachung durch die Anwendung (12).

3. Asynchroner Datenübertragungskanal (20) nach Anspruch 2, wobei die Registerdatei (26) eine Zieladresse, eine zu übertragende Datenlänge, eine Länge jedes zu übertragenden Datenpakets sowie eine Übertragungsrichtung aufweist.

BEST AVAILABLE COPY

4. Asynchroner Datenübertragungskanal (20) nach Anspruch 3, ferner mit einer physischen Busschnittstelle, die so konfiguriert ist, daß sie die Busstruktur (58) derart verbindet, daß Daten auf der Busstruktur (58) plaziert und Daten aus der Busstruktur (58) gewonnen werden.

5. Asynchroner Datenübertragungskanal (20) nach Anspruch 4, wobei die für die Vollendung der Datenübertragungsoperation erforderlichen Transaktionen für einen ansteigenden Adreßbereich erzeugt werden.

6. Asynchroner Datenübertragungskanal (20) nach Anspruch 4, wobei die für die Vollendung der Datenübertragungsoperation erforderlichen Transaktionen für eine feste Adresse erzeugt werden.

7. Asynchroner Datenübertragungskanal (20) nach Anspruch 4, wobei die Registerdatei (26) ferner einen Paketzählerwert aufweist, der die Anzahl der Pakete darstellt, die für die Übertragung verbleiben, wobei der Paketzählerwert nach der Übertragung jedes Datenpakets dekrementiert wird.

8. Asynchroner Datenübertragungskanal (20) nach Anspruch 7, wobei die Anwendung (12) den Paketzählerwert automatisch erhöht, indem ein vorbestimmtes Feld in der Registerdatei (26) beschrieben wird.

9. Asynchroner Datenübertragungskanal (20) nach Anspruch 8, wobei es sich bei der Busstruktur (58) um eine IEEE-1394-Busstruktur handelt.

10. Verfahren zur Verwaltung einer Schreib-Datenübertragungsoperation zwischen einer Anwendung (12) und

BEST AVAILABLE COPY

einem mit einer Busstruktur (58) gekoppelten Knoten, wobei das Verfahren die folgenden Schritte umfaßt:

(a) Empfangen von Anweisungen in Bezug auf eine Schreib-Datenübertragungsoperation von der Anwendung (12);

(b) Speichern der genannten Anweisungen in einer Registerdatei (26);

(c) Erlangen eines Datenpakets von der Anwendung;

(d) Erzeugen eines Headers für die Datenübertragungsoperation, wobei der Header ohne unmittelbare Prozessorsteuerung oder Überwachung durch die Anwendung erzeugt wird;

(e) Hinzufügen des Headers zu dem Datenpaket, wobei der Header eine Zieladresse für das Datenpaket aufweist; und

(f) Übertragen des Datenpakets, einschließlich des Headers, auf die Busstruktur (58).

11. Verfahren nach Anspruch 10, wobei die Anweisungen eine Zieladresse, eine Länge der zu übertragenden Daten, eine Länge jedes zu übertragenden Datenpakets sowie einen Paketzählerwert aufweisen, der die Anzahl der zu übertragenden Pakete darstellt.

12. Verfahren nach Anspruch 11, wobei die Registerdatei (26) als Vorlage für die Erzeugung des Headers und der Transaktionen verwendet wird, die erforderlich sind, um ein Datenpaket auf die Busstruktur (58) zu schreiben, und zwar ohne unmittelbare Prozessorsteuerung oder Überwachung durch die Anwendung (12).

13. Verfahren nach Anspruch 12, wobei das Verfahren ferner die folgenden Schritte umfaßt:

(g) Erweiterung der Zieladresse um die Länge eines Datenpakets;

BEST AVAILABLE COPY

(h) Dekrementieren des Paketzählerwertes; und

(i) Wiederholen der Schritte (b) - (h) für jedes zu übertragende Datenpaket, bis der Paketzählerwert gleich Null ist.

14. Verfahren nach Anspruch 13, wobei das Datenpaket aus dem Datenspeicherpuffer (32) gewonnen wird, der durch die Anwendung (12) geladen wird.

15. Verfahren zur Verwaltung einer Lese-Datenübertragungsoperation zwischen einer Anwendung (12) und einem mit einer Busstruktur (58) gekoppelten Knoten, wobei das Verfahren die folgender Schritte umfaßt:

(a) Empfangen von Anweisungen in Bezug auf eine Lese-Datenübertragungsoperation von der Anwendung (12);

(b) Speichern der genannten Anweisungen in einer Registerdatei (26);

(c) Erzeugen einer Transaktion, die für die Anforderung erforderlich ist, daß ein Datenpaket von dem Knoten auf der Busstruktur (58) plaziert wird, wobei die Transaktion ohne direkte Prozessorsteuerung oder Überwachung durch die Anwendung (12) erzeugt wird;

(d) Erlangen des Datenpakets von der Busstruktur (58);

(e) Auslesen des Datenpakets ohne Headerdaten in die Anwendung (12).

16. Verfahren nach Anspruch 15, wobei die Anweisungen eine Zieladresse aufweisen, die eine Startadresse in dem Knoten darstellt, von der aus die Daten übermittelt werden, eine Länge der zu übertragenden Daten, eine Länge jedes zu übertragenden Datenpakets und einen Paketzählerwert, der einer Anzahl zu übertragender Pakete entspricht.

BEST AVAILABLE COPY

17. Verfahren nach Anspruch 16, wobei die Registerdatei (26) als Vorlage für die Erzeugung der zum Lesen eines Datenpakets aus dem Knoten erforderlichen Transaktion und des Headers verwendet wird, und zwar ohne unmittelbare Prozessorsteuerung oder Überwachung.

18. Verfahren nach Anspruch 17, wobei das Verfahren ferner die folgenden Schritte umfaßt:

(g) Erweitern der Zieladresse um die Länge eines Datenpakets;

(h) Dekrementieren des Paketzählerwertes; und

(i) Wiederholen der Schritte (b) bis (h) für jedes zu übertragende Datenpaket, bis der Paketzählerwert gleich Null ist.

19. Verfahren nach Anspruch 18, wobei das Datenpaket über einen Datenspeicherpuffer (32) an die Anwendung (12) vorgesehen wird.

BEST AVAILABLE COPY

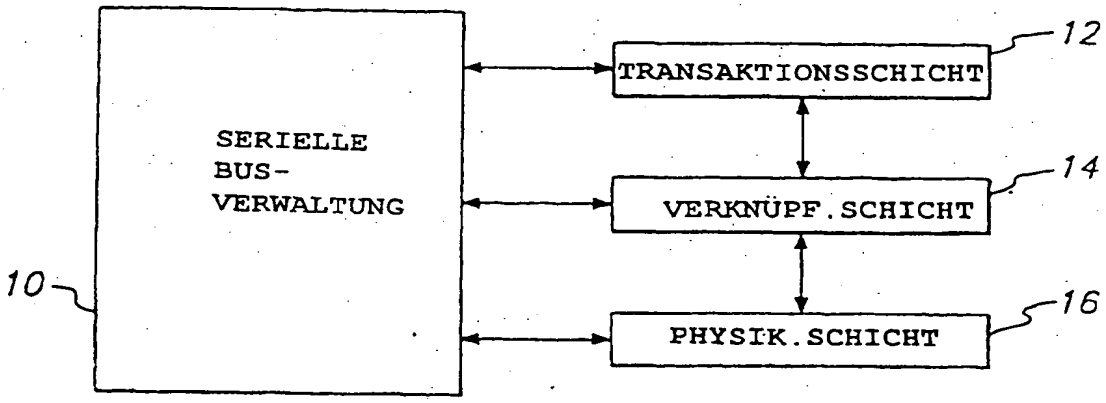

FIG. 1
(STAND DER TECHNIK)

BEST AVAILABLE COPY

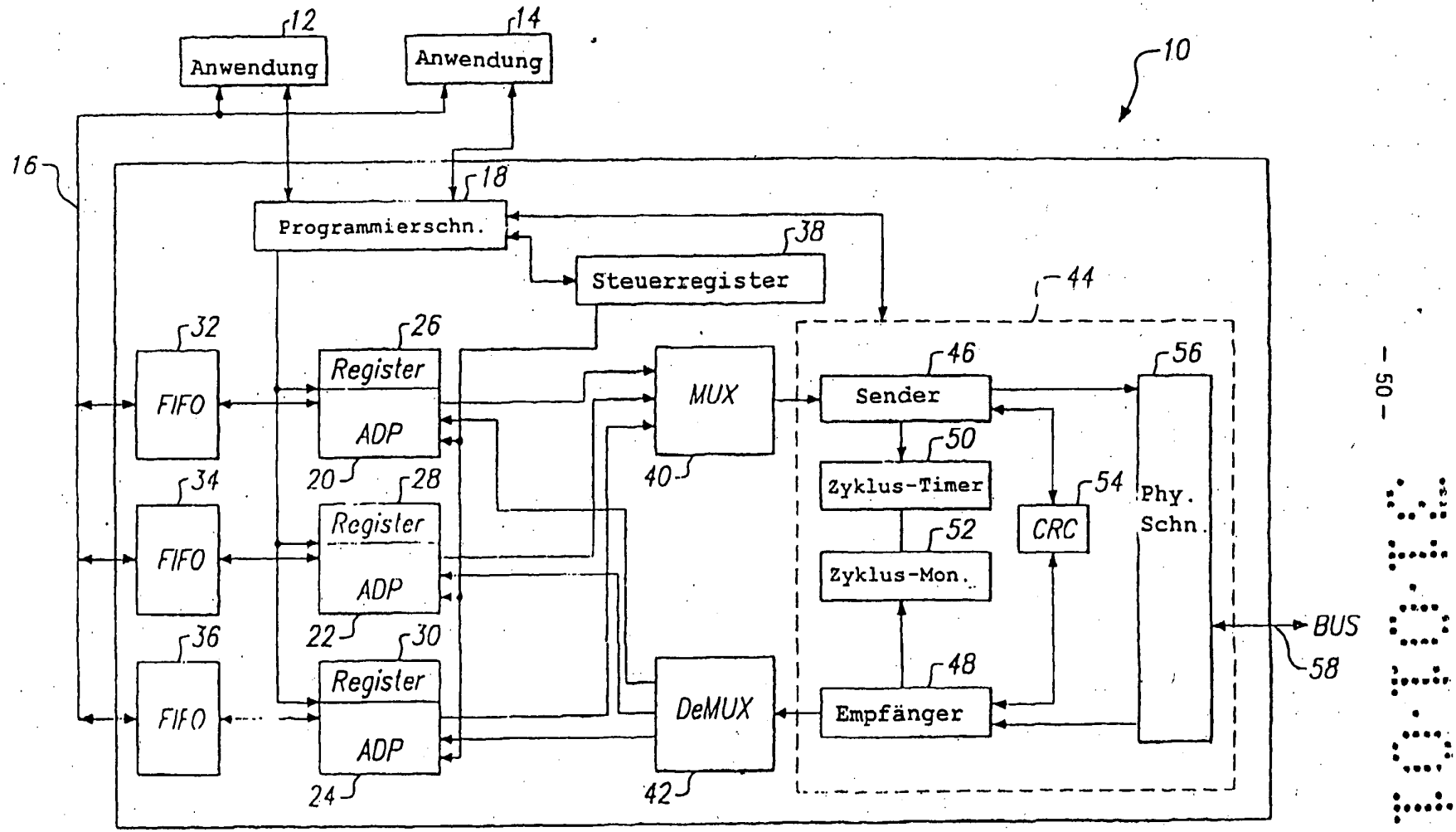

FIG. 2

BEST AVAILABLE COPY

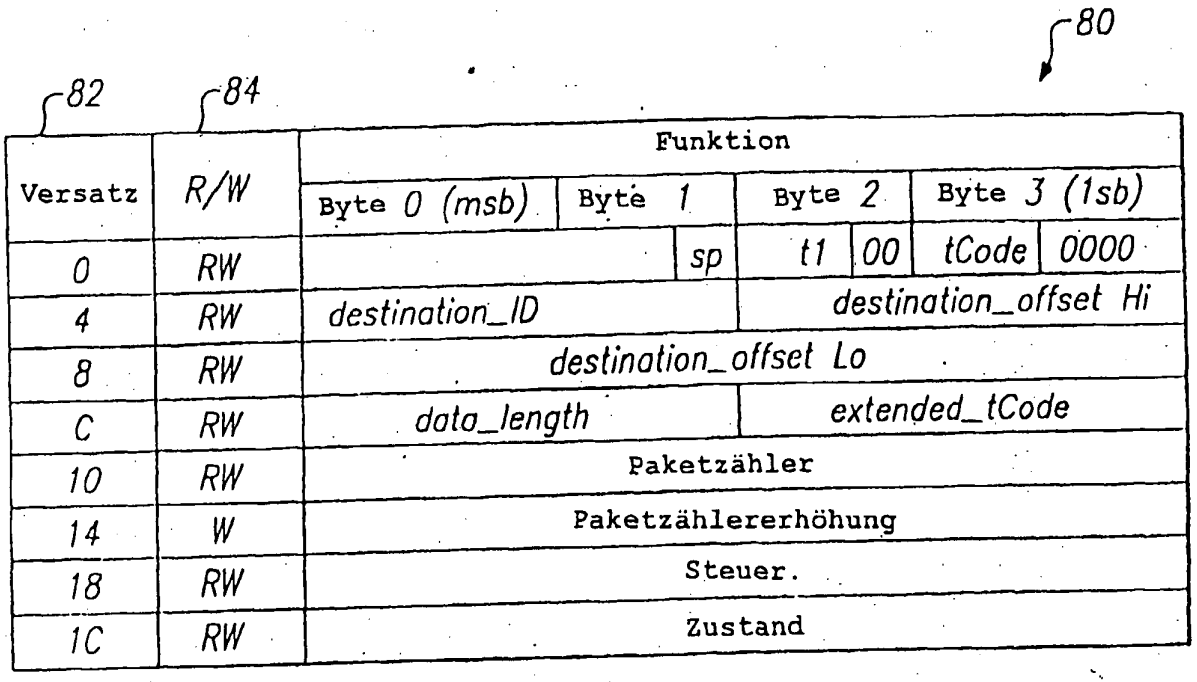

Abbildung der ADP-Steuerregister

FIG. 3

BEST AVAILABLE COPY